3. (Currently amended) The system of claim 1 wherein two or more interrupts or exceptions may be ~~mapped~~ directed to one stream.

4. (Currently amended) The system of claim 1 wherein ~~mapping~~ directing of interrupts to streams is static and determined at processor design.

5. (Currently amended) The system of claim 1 wherein ~~mapping~~ directing of interrupts and exceptions is programmable.

6. (Currently amended) The system of claim 5 wherein ~~mapping~~ directing is programmed in a data store, and the interrupt logic refers to the data store for ~~mapping~~ directing data to relate received interrupts or exceptions to streams.

7. (Currently amended) The system of claim 1 wherein ~~mapping~~ directing is conditional and dynamic, the interrupt logic executing an algorithm sensitive to variables to determine the mapping.

8. (Original) The system of claim 1 wherein the interrupts are external interrupts generated by devices external to the processor.

9. (Original) The system of claim 1 wherein the interrupts are software interrupts generated by active streams.

10. (Currently amended) The system of claim 6 wherein the data storage further comprises a mask for enabling/disabling execution of ~~mapped~~ directed interrupts or exceptions.

11. (Currently amended) The system of claim 1 wherein, after ~~mapping~~ directing is determined for a detected interrupt or exception the one or more streams are interrupted by the interrupt logic.

12. (Original) The system of claim 11 wherein an interrupted stream acknowledges the interrupt, and is vectored to a service routing by the interrupt logic.

13. (Original) The system of claim 12 wherein two or more streams are interrupted by one interrupt or exception, and wherein the interrupt logic delays vectoring any stream to a service routine until all interrupted streams acknowledge the interrupt.

14. (Original) The system of claim 13 wherein two streams acknowledging the same interrupt are vectored to different service routines by the interrupt logic.

15. (Currently Amended) A method for processing interrupts in a multi-stream processor comprising steps of:

(a) detecting an interrupt or exception and passing the detected interrupt or exception to interrupt logic; and

(b) ~~mapping the interrupt or exception to~~ directing one or more specific streams of the multi-stream processor to process a specific inturrupt or exception at the time of their detection.

16 (Currently amended) The method of claim 15 wherein, in step (b), the interrupt or exception may be ~~mapped~~ directed to two or more streams.

17. (Currently amended) The method of claim 15 wherein, in step (a) two or more interrupts or exceptions are detected, and in step (b), the two or more interrupts or exceptions are ~~mapped~~ directed to one stream.

18. (Currently amended) The method of claim 15 wherein ~~mapping~~ direction of interrupts to streams is static and determined at processor design.

19. (Currently amended) The method of claim 15 wherein ~~mapping~~ directing of interrupts and exceptions is programmable.

20. (Currently amended) The method of claim 19 wherein ~~mapping~~ directing is programmed in a data store, and the interrupt logic refers to the data store for mapping data to relate received interrupts or exceptions to streams.

21. (Currently amended) The method of claim 15 wherein ~~mapping~~ directing is conditional and dynamic, the interrupt logic executing an algorithm sensitive to variables to determine the mapping.

22. (Original) The method of claim 15 wherein the interrupts are external interrupts generated by devices external to the processor.

23. (Original) The method of claim 15 wherein the interrupts are software interrupts generated by active streams.

24. (Currently amended) The method of claim 20 wherein the data storage further comprises a mask for enabling/disabling execution of ~~mapped~~ directed interrupts or exceptions.

25. (Currently amended) The method of claim 15 further comprising a step for, after ~~mapping~~ directing is determined for a detected interrupt or exception, the one or more streams are interrupted by the interrupt logic.

26. (Original) The method of claim 25 comprising a further step for vectoring an interrupted stream to a service routine after the interrupted stream acknowledges the interrupt.

27. (Original) The method of claim 26 wherein two or more streams are interrupted by one interrupt or exception, and wherein the interrupt logic delays vectoring any stream to a service routine until all interrupted streams acknowledge the interrupt.

28. (Original) The method of claim 27 wherein two streams acknowledging the same interrupt are vectored to different service routines by the interrupt logic.

29. (Currently Amended) A computing system comprising:

input apparatus for acquiring data to be processed;

memory elements for storing data and executable code for controlled use;

a multi-streaming processor having a plurality of streams for streaming one or more instruction threads; and

interrupt handling logic;

characterized in that through the interrupt logic specific interrupts or exceptions are detected and at the time of their detection ~~mapped to one or more specific streams~~ a specific stream is directed to process the specific interrupt or exception.

30. (Currently amended) The system of claim 28 wherein one interrupt or exception may be ~~mapped~~ directed to two or more streams.

31. (Currently amended) The system of claim 28 wherein two or more interrupts or exceptions may be ~~mapped~~ directed to one stream.

32. (Currently amended) The system of claim 28 wherein ~~mapping~~ directing of interrupts to streams is static and determined at processor design.

33. (Currently amended) The system of claim 28 wherein ~~mapping~~ directing of interrupts and exceptions is programmable.

34. (Currently amended) The system of claim 33 wherein ~~mapping~~ directing is programmed in a data store, and the interrupt logic refers to the data store for mapping data to relate received interrupts or exceptions to streams.

35. (Currently amended) The system of claim 28 wherein ~~mapping~~ directing is conditional and dynamic, the interrupt logic executing an algorithm sensitive to variables to determine the mapping.

36. (Original) The system of claim 28 wherein the interrupts are external interrupts generated by devices external to the processor.

37. (Original) The system of claim 28 wherein the interrupts are software interrupts generated by active streams.

38. (Original) The system of claim 34 wherein the data storage further comprises a mask for enabling/disabling execution of mapped interrupts or exceptions.

39. (Currently amended) The system of claim 28 wherein, after ~~mapping~~ directing is determined for a detected interrupt or exception the one or more streams are interrupted by the interrupt logic.

40. (Original) The system of claim 39 wherein an interrupted stream acknowledges the interrupt, and is vectored to a service routing by the interrupt logic.

41. (Original) The system of claim 40 wherein two or more streams are interrupted by one interrupt or exception, and wherein the interrupt logic delays vectoring any stream to a service routine until all interrupted streams acknowledge the interrupt.

42. (Original) The system of claim 41 wherein two streams acknowledging the same interrupt are vectored to different service routines by the interrupt logic.